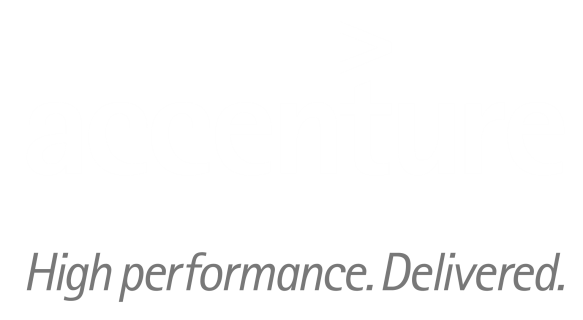
Axure2JSF



**ARTL for Java**

Java Technical Good

Install MongoDB on windows

*Quick Start and User Guide*

*June 2015 version 1.00*

*March, 2013*

*Version 1.0*

Table of Contents

[1 MongoDB ( NoSQL database ) 3](#_Toc350505671)

[1.1 Introduction 3](#_Toc350505672)

[1.2 Purpose of this document 3](#_Toc350505673)

[1.3 Document structure 3](#_Toc350505674)

[2 MongoDB Data Model and Deployment Architecture 4](#_Toc350505675)

[2.1 RDBMS model vs MongoDB data model 4](#_Toc350505676)

[2.2 How does mongodb works ? 5](#_Toc350505680)

[2.3 Deployment Architecture 5](#_Toc350505681)

[3 Installing MongoDB on windows 6](#_Toc350505685)

[3.1 Download MongoDB 6](#_Toc350505686)

[3.2 Review MongoDB folder 6](#_Toc350505687)

[3.3 Configuration File 7](#_Toc350505688)

[3.4 Run MongoDB server 7](#_Toc350505686)

[3.5 Connect to MongoDB 7](#_Toc350505687)

[3.6 MongoDB as Windows Service 8](#_Toc350505688)

[4 MongoDB use cases 9](#_Toc350505685)

[5 Document Control 10](#_Toc350505689)

[4.1 Change History 10](#_Toc350505690)

[4.2 Open Issues 10](#_Toc350505691)

# mongodb

## Introduction

MongoDB is a document database that provides high performance, high availability, and easy scalability.

* *Flexible Data Model :*

MongoDB’s document data model makes it easy for you to store data of any structure and dynamically modify the schema.Embedded documents and arrays reduce need for joins.Dynamic schema makes polymorphism easier.

* *Highly Scalable :*

Scale up or scale out horizontally, from a single server to thousands of nodes. Deploy in the cloud and across multiple data centers.

* *Robust Operational Tools :*

MongoDB Management Service and Ops Manager allow you to deploy, monitor, back up and scale MongoDB with ease.

* *Expressive Query Language :*

MongoDB’s query language provides varied field-level operators, data types and in-place updates. Drivers for just about any programming language make it intuitive to use.

* *High Performance :*

Embedding makes reads and writes fast. ◦Indexes can include keys from embedded documents and arrays.

* *High Availability :*

Replicated servers with automatic master failover.

## Purpose of this document

This document shows MongoDB’s data model and usage scenarios.

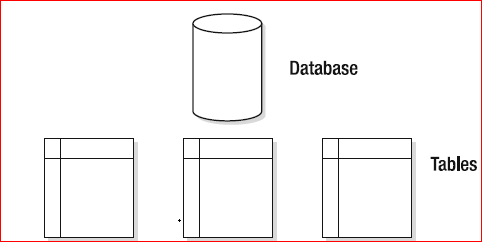
## Document structure

Section 2 presents an overview about MongoDB’s data model. Section 3 shows the installation steps for MongoDB on windows. Section 4 shows a usage example.

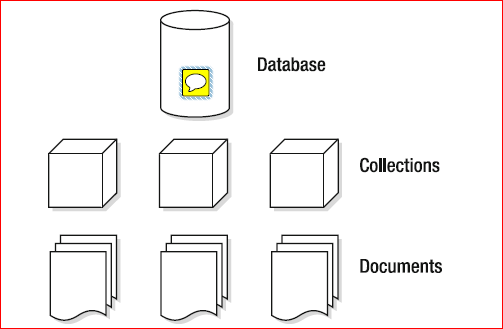
# Mongodb data model

## RDBMS Model V/S MongoDB Model

Typical RDBMS model



MongoDB Database model



## How does MongoDB works?

A MongoDB deployment hosts a number of databases. A database holds a set of collections. A collection holds a set of documents. A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection’s documents may hold different types of data.

Developers access documents through rich, idiomatic drivers available in all popular programming languages. Documents map naturally to the objects in modern languages, which allows developers to be extremely productive. Typically, there’s no need for an ORM layer.

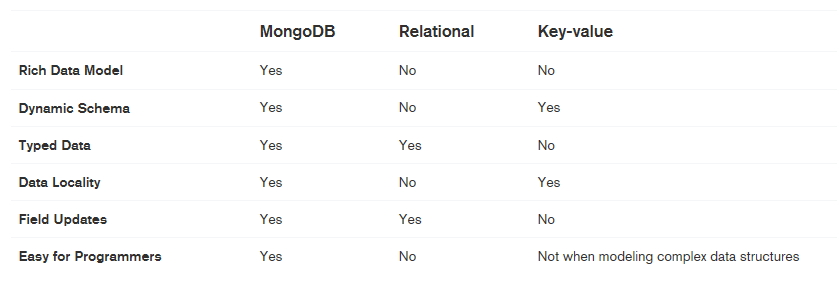
MongoDB provides auto-sharding for horizontal scale out. Native replication and automatic leader election supports high availability across racks and data centers. And MongoDB makes extensive use of RAM, providing in-memory speed and on-disk capacity.

Unlike most NoSQL databases, MongoDB provides comprehensive secondary indexes, including geospatial and text search, as well as extensive security and aggregation capabilities. MongoDB provides the features you need to develop the majority of the new applications your organization develops today.

## Deployment Architectures

Although MongoDB supports a “standalone” or single-instance operation, production MongoDB deployments are distributed by default. Replica sets provide high performance replication with automated failover, while sharded clusters make it possible to partition large data sets over many machines transparently to the users. MongoDB users combine replica sets and sharded clusters to provide high levels redundancy for large data sets transparently for applications.

How does the MongoDB data model stack up to relational databases and key-value stores? Take a look at the chart below:



# installing mongodb on windows

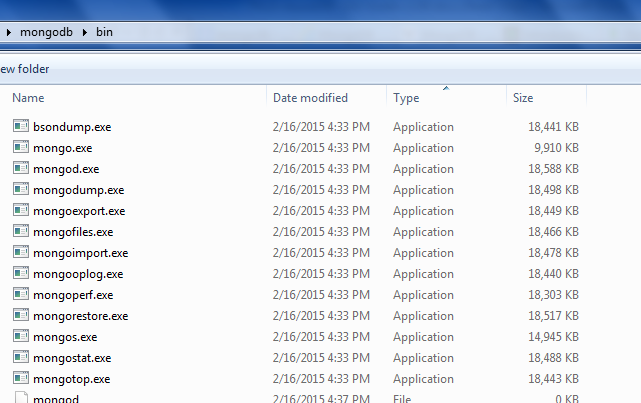
## Download MongoDB

Download MongoDB from official MongoDB website. Choose Windows 32 bits or 64 bits. Unzip, extracts to your prefer location, for example : d:\mongodb

## Review MongoDB folder

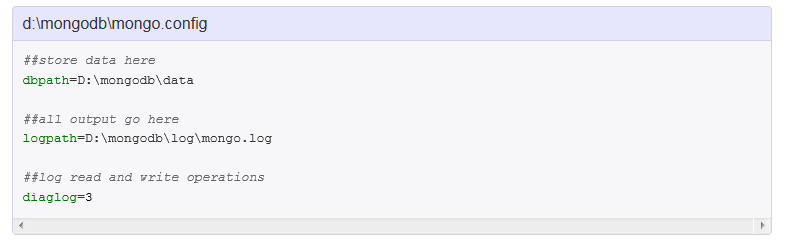
In MongoDB, it contains only 10+ executable files (exe) in the bin folder. This is true, and that are the required files to MongoDB, it’s really hard to believe for a developer like me who come from a relation database background.

Figure : Files under $MongoDB/bin folder



## Configuration File

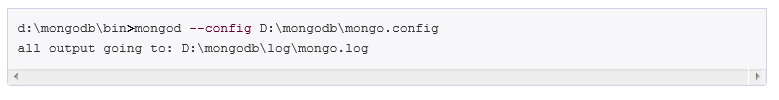
Create a MongoDB config file, it’s just a text file, for example : d:\mongodb\mongo.config



**Note:**  
MongoDB need a folder (data directory) to store its data. By default, it will store in “C:\data\db“, create this folder manually. MongoDB won’t create it for you. You can also specify an alternate data directory with --dbpath option.

## Run MongoDB server

Use mongod.exe --config d:\mongodb\mongo.config to start MongoDB server.



## 3.5 Connect to MongoDB

Uses mongo.exe to connect to the started MongoDB server.



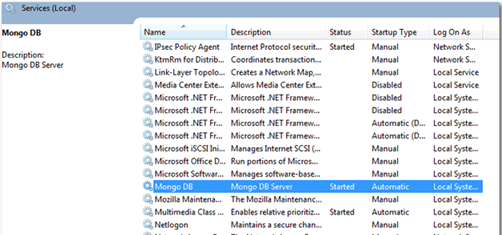
## 3.6 MongoDB as Windows Service

Add MongoDB as Windows Service, so that MongoDB will start automatically following each system restart.

Install as Windows Service with --install.



A Windows service named “MongoDB” is created.



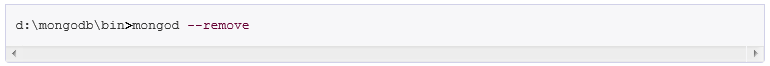
To start MongoDB Service



To stop MongoDB Service



To remove MongoDB Service



# Mongodb usecase

* *You Expect a High Write Load*

MongoDB by default prefers high insert rate over transaction safety. If you need to load tons of data lines with a low business value for each one, MongoDB should fit. Don't do that with $1M transactions recording or at least in these cases do it with an extra safety.

* *You need High Availability in an Unreliable Environment (Cloud and Real Life)*

Setting replicaSet (set of servers that act as Master-Slaves) is easy and fast. Moreover, recovery from a node (or a data center) failure is instant, safe and automatic

* *You need to Grow Big (and Shard Your Data)*

Databases scaling is hard (a single MySQL table performance will degrade when crossing the 5-10GB per table). If you need to partition and shard your database, MongoDB has a built in easy solution for that.

* *Your Data is Location Based*

MongoDB has built in spacial functions, so finding relevant data from specific locations is fast and accurate.

* *Your Data Set is Going to be Big (starting from 1GB) and Schema is Not Stable*

Adding new columns to RDBMS can lock the entire database in some database, or create a major load and performance degradation in other. Usually it happens when table size is larger than 1GB (and can be major pain for a system like BillRun that is described bellow and has several TB in a single table). As MongoDB is schema-less, adding a new field, does not effect old rows (or documents) and will be instant. Other plus is that you do not need a DBA to modify your schema when application changes.

* *You Don't have a DBA*

If you don't have a DBA, and you don't want to normalize your data and do joins, you should consider MongoDB. MongoDB is great for class persistence, as classes can be serialized to JSON and stored AS IS in MongoDB. Note: If you are expecting to go big, please notice that you will need to follow some best practices to avoid pitfalls.

# Document Control

## Change History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Approver** | **Comment** |
| 1 | 15/06/2015 | 1 | Ashish Agarwal |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Open Issues

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Action** | **Owner** | **Status** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |